



PCT/FR 2004/003353

REC'D 11 MAR 2005
WIPO PCT

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le 27 DEC. 2004

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brevets

A handwritten signature in black ink, appearing to read 'Martine PLANCHE'.

Martine PLANCHE

DOCUMENT DE PRIORITÉ

PRÉSENTÉ OU TRANSMIS
CONFORMÉMENT À LA
RÈGLE 17.1.a) OU b)

SIEGE
26 bis, rue de Saint-Petersbourg
75800 PARIS cedex 08
Téléphone : 33 (0)1 53 04 53 04
Télécopie : 33 (0)1 53 04 45 23
www.inpi.fr



REQUÊTE EN DÉLIVRANCE

DB 540 e W / 210502

REMISSION DES PIÈCES		Réervé à l'INPI	Cet imprimé est à remplir lisiblement à l'encre noire	
DATE	24 DEC. 2003		DB 540 e W / 21	
LIEU	09			
N° D'ENREGISTREMENT	0315487		NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE	
NATIONAL ATTRIBUÉ PAR L'INPI		À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE		
DATE DE DÉPÔT ATTRIBUÉE PAR L'INPI		24-12-2003		CABINET MOUTARD BP 513 78005 VERSAILLES CEDEX FRANCE
Vos références pour ce dossier (facultatif) TRUSB0020				
Confirmation d'un dépôt par télécopie			<input type="checkbox"/> N° attribué par l'INPI à la télécopie	
2 NATURE DE LA DEMANDE			Cochez l'une des 4 cases suivantes	
Demande de brevet			<input checked="" type="checkbox"/>	
Demande de certificat d'utilité			<input type="checkbox"/>	
Demande divisionnaire			<input type="checkbox"/>	
Demande de brevet initiale ou demande de certificat d'utilité initiale			N°	Date
Transformation d'une demande de brevet européen Demande de brevet initiale			N°	Date
			N°	Date
3 TITRE DE L'INVENTION (200 caractères ou espaces maximum)				
PROCEDE DE MISE A JOUR D'APPLICATIONS POUR CARTE A PUCE.				
4 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE FRANÇAISE			Pays ou organisation Date [] N°	
			Pays ou organisation Date [] N°	
			Pays ou organisation Date [] N°	
			<input type="checkbox"/> S'il y a d'autres priorités, cochez la case et utilisez l'imprimé «Suite»	
5 DEMANDEUR (Cochez l'une des 2 cases)			<input checked="" type="checkbox"/> Personne morale	<input type="checkbox"/> Personne physique
Nom ou dénomination sociale			TRUSTED LOGIC	
Prénoms				
Forme juridique			Société Anonyme	
N° SIREN			[4 2 1 4 8 3 4 1 3]	
Code APE-NAF			[7 2 1 Z]	
Domicile ou siège	Rue	5 rue du Bailliage		
	Code postal et ville	[7 8 0 0 0] Versailles		
Nationalité	Pays	FRANCE Française		
N° de téléphone (facultatif)	N° de télécopie (facultatif)			
Adresse électronique (facultatif)				
<input type="checkbox"/> S'il y a plus d'un demandeur, cochez la case et utilisez l'imprimé «Suite»				

S'il y a plus d'un demandeur, cochez la case et utilisez l'imprimé « Suite »

Remplir impérativement la 2^{ème} page

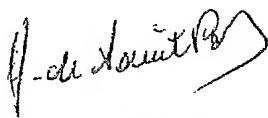
**BREVET D'INVENTION
CERTIFICAT D'UTILITÉ**

REQUÊTE EN DÉLIVRANCE
page 2/2

BR2

REMISE DES PIÈCES	
DATE	Réervé à l'INPI
LIEU	24 DEC. 2003
N° D'ENREGISTREMENT	0315487
NATIONAL ATTRIBUÉ PAR L'INPI	

DB 540 W / 210502

6 MANDATAIRE (s'il y a lieu)	
Nom _____ Prénom _____ Cabinet ou Société _____	
N °de pouvoir permanent et/ou de lien contractuel _____	
Adresse	Rue _____
	Code postal et ville _____
	Pays _____
N ° de téléphone (facultatif)	0130837979
N ° de télécopie (facultatif)	0130837978
Adresse électronique (facultatif)	asp@moutard.fr
7 INVENTEUR (S)	
Les demandeurs et les inventeurs sont les mêmes personnes <input type="checkbox"/> Oui <input checked="" type="checkbox"/> Non : Dans ce cas remplir le formulaire de Désignation d'inventeur(s)	
8 RAPPORT DE RECHERCHE	
Établissement immédiat ou établissement différé <input checked="" type="checkbox"/> <input type="checkbox"/>	
Paiement échelonné de la redevance (en deux versements) <input type="checkbox"/> <input checked="" type="checkbox"/>	
9 RÉDUCTION DU TAUX DES REDEVANCES	
Uniquement pour les personnes physiques <input type="checkbox"/> Requise pour la première fois pour cette invention (joindre un avis de non-imposition) <input type="checkbox"/> Obtenu antérieurement à ce dépôt pour cette invention (joindre une copie de la décision d'admission à l'assistance gratuite ou indiquer sa référence) : AG _____	
10 SÉQUENCES DE NUCLEOTIDES ET/OU D'ACIDES AMINÉS	
<input type="checkbox"/> Cochez la case si la description contient une liste de séquences	
Le support électronique de données est joint <input type="checkbox"/> <input type="checkbox"/>	
La déclaration de conformité de la liste de séquences sur support papier avec le support électronique de données est jointe	
Si vous avez utilisé l'imprimé « Suite », indiquez le nombre de pages jointes	
11 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE (Nom et qualité du signataire)	
A. de Saint Palais N° 94 0306 	
VISA DE LA PRÉFECTURE OU DE L'INPI	
M. ROCHE	

- 10 La présente invention concerne un procédé de mise à jour d'applications pour carte à puce.

Elle a plus particulièrement pour objet de permettre le chargement d'une nouvelle version d'une application dans une carte à puce tout en conservant les 15 données utilisées par les précédentes versions de l'application.

La plupart des systèmes de cartes à puce actuelles utilisent des machines virtuelles, la plus populaire d'entre elle étant celle de « Java Card » (marque déposée par la Société Sun Microsystems). Dans ce système, les données 20 persistantes des applications sont stockées comme des objets, et en particulier comme des instances des classes définies dans les programmes chargés dans la carte. La nature persistante de ces objets rend donc les programmes actifs en permanence dans la carte, à l'inverse de ce qui se produit dans les systèmes classiques (stations de travail, ordinateurs de bureau). Cela pose un problème 25 particulier pour la mise à jour des programmes, puisqu'il n'est pas possible de profiter d'un moment où le programme n'est pas actif pour le mettre à jour.

De plus, la plupart des plates-formes utilisent un format de code binaire optimisé (format dit « CAP File » dans le cas de « Java Card »). Ce format 30 permet de ne charger sur une carte que les informations strictement nécessaires à l'exécution du programme. En particulier, il ne contient pas

toujours les informations nécessaires pour effectuer une mise à jour de l'application.

Enfin, une carte à puce est un contexte d'exécution très particulier, dans la mesure où elle est diffusée en de très nombreux exemplaires, et elle contient souvent des informations confidentielles. En particulier, il est très difficile d'envisager de mettre à jour une application en effaçant l'application existante et ses données, en rechargeant la nouvelle application, puis en rechargeant les données. En effet, les données d'initialisation des applications sont généralement très sensibles, et ne doivent absolument pas être manipulées en dehors des sites de production des cartes, sous des conditions de sécurité très contrôlées.

La nature des problèmes rencontrés dépend fortement des caractéristiques de la mise à jour souhaitée. Pour la mise à jour d'une application, les problèmes sont les suivants :

- Mise à jour de code simple.

Le problème est ici de corriger des défauts du programme, sans en modifier la structure. Il s'agit donc simplement de modifier la définition de certaines méthodes, et éventuellement de rajouter de nouvelles méthodes, voire de nouvelles classes (en dehors de la hiérarchie existante).

- Mise à jour de code avec modification de la hiérarchie de classes.

Le problème est ici de corriger des défauts structurels du programme, nécessitant une modification de la hiérarchie de classes (habituellement par insertion d'une classe au milieu d'une hiérarchie existante pour prendre en compte un comportement spécifique).

- o Mise à jour avec modification de la structure de l'objet.

Le problème est ici de corriger un défaut nécessitant le stockage d'informations supplémentaires, et donc l'ajout de champs de données dans des classes existantes. Ce problème est plus complexe dans la mesure où les 5 objets existants doivent être modifiés pour prendre en compte les modifications.

Dans certains cas, l'application à mettre à jour est accessible depuis d'autres applications, en particulier quand l'application exporte des interfaces 10 partageables et quand l'application est en fait une librairie. D'un point de vue purement technique, de telles mises à jour posent les mêmes problèmes que les mises à jour simples d'applications, à la différence près que les mises à jour doivent être appliquées à toutes les applications qui importent les fonctions modifiées.

15

L'invention a donc plus particulièrement pour but la réalisation d'un mécanisme de chargement d'une nouvelle version d'une application, qui remplace une application déjà présente sur une carte avec des garanties en matière de sécurité et de continuité de fonctionnement, ce mécanisme 20 permettant donc de modifier des applications sans interrompre leur fonctionnement.

Elle se propose d'aboutir à ce résultat en utilisant une option spécifique de commande de chargement et un format de chargement compatible avec celui 25 défini dans la spécification « Java Card ».

En vue de parvenir à ces résultats, elle propose, d'une façon générale, un procédé pour le chargement sur un dispositif informatique d'une nouvelle version d'une application dans un langage de programmation à objets et 30 permettant, notamment, l'introduction de classes supplémentaires, la

modification de la hiérarchie de classes et la définition de champs et de méthodes supplémentaires.

Selon l'invention, ce procédé consiste à :

5

- calculer préalablement à ce chargement une information permettant d'établir la correspondance entre les classes de l'ancienne version de l'application et les classes de la nouvelle version de l'application ;

10 - calculer préalablement à ce chargement une information permettant d'établir la correspondance entre les identificateurs des champs statiques de l'ancienne version de l'application des identificateurs des champs statiques de la nouvelle version de l'application ;

- associer lesdites informations de correspondance à la nouvelle version de l'application chargée sur le dispositif ;

15 - utiliser lesdites informations de correspondance pour modifier les objets de façon à ce qu'ils pointent vers les classes de la nouvelle version de l'application et qu'ils utilisent les nouveaux identificateurs des champs statiques de la nouvelle version de l'application.

20 Avantageusement, lesdites informations de correspondance pourront consister en des tables de correspondance.

Elles pourront être omises dans le cas où les modifications des objets ne sont pas nécessaires, par exemple, et de façon non limitative, quand aucune classe supplémentaire n'est ajoutée dans la nouvelle version de l'application ou quand les classes ajoutées ne modifient pas la hiérarchie de classes.

Le procédé selon l'invention pourra comprendre l'exécution de méthodes de mise à jour des données de l'application après l'installation de la nouvelle 30 version de l'application.

Un mode d'exécution de l'invention sera décrit ci-après, à titre d'exemple non limitatif, avec référence aux dessins annexés dans lesquels :

5 Les figures 1a et 1b sont des tableaux comparatifs montrant la hiérarchie de classes de l'application, dans sa version originale (figure 1a) et dans sa nouvelle version (figure 1b) ;

10 La figure 2 est une table de correspondance donnant, pour chaque classe de l'application originale, l'index de la classe correspondante dans la nouvelle application ;

15 Les figures 3a et 3b sont des tableaux comparatifs montrant une hiérarchie avec la propriété recherchée, hiérarchie originale (figure 3a) et nouvelle hiérarchie (figure 3b) ;

20 La figure 4 est une table de correspondance entre des tables de correspondance (table originale/nouvelle table) correspondant à la hiérarchie illustrée figure 3b ;

25 La figure 5 est une table de correspondance du type de celle de la figure 4 pour les champs statiques ;

Les figures 6 à 8 sont des représentations schématiques illustrant les états d'une carte à puce, avant mise à jour (figure 6), après chargement de la nouvelle application (figure 7) et après modification des objets (figure 8).

La mise en oeuvre de l'invention se fait en plusieurs étapes :

- 30 • Préparation du fichier de chargement.
• Chargement du fichier et édition de liens.

- Mise à jour des objets de l'application.
- Exécution d'une méthode spécifique de mise à jour.

5 Le fichier de chargement doit contenir des informations spécifiques qui permettent d'établir la correspondance avec une ou plusieurs versions antérieures de l'application. Afin de générer le fichier de mise à jour, il est nécessaire de disposer des objets suivants :

- Tous les fichiers de classe de l'application originale.
- 10 ◦ Le fichier de chargement (« CAP File » dans le cas de « Java Card ») de l'application originale.
- Tous les fichiers de classe de la nouvelle version de l'application.
- Tous les fichiers d'export requis pour construire la nouvelle version de l'application.

15

Si le fichier de chargement de l'application originale comprend des informations de typage et de nommage des classes, il n'est alors pas indispensable de disposer des fichiers de classe de l'application originale.

20 Les données d'entrées doivent respecter les contraintes suivantes :

- Pour chaque élément de l'application originale, il existe dans la nouvelle version de l'application un élément équivalent (de même nom et de même type).
- 25 ◦ Si l'application originale exporte une interface externe aux autres applications, cette interface externe doit être inchangée dans la nouvelle version de l'application.
- Les fichiers d'export utilisés dans la nouvelle version de l'applications sont binaires-compatibles (c'est-à-dire qu'ils peuvent être reliés sans modification aux autres parties de l'application initiale) avec ceux utilisés dans l'application originale. Dans le cas de « Java Card », il s'agit de ceux

30

qui sont listés dans le composant d'import du fichier de chargement original.

Le fichier de chargement généré est un fichier de chargement normal, qui peut
5 donc être chargé sur n'importe quelle carte. Ce fichier contient un composant
optionnel avec les informations suivantes pour chaque version précédente de
l'application prise en considération :

- le numéro de cette version,
- 10 - une table qui établit une correspondance entre chaque classe ou interface
définie dans l'ancienne version et la nouvelle version de cette classe ou
interface dans la nouvelle version de l'application,
- une table qui établit une correspondance entre l'identifiant de chaque champ
statique de l'ancienne version et le nouvel identifiant de ce champ dans la
15 nouvelle version.

Ces informations supplémentaires ne sont nécessaires que pour les opérations
de mise à jour. Le même fichier binaire peut donc servir pour le chargement
de l'application et pour la mise à jour d'applications.

20 Dans un premier exemple, les hiérarchies de classe de l'application dans sa
version originale et dans sa nouvelle version sont montrées sur les figures 1a
et 1b. La hiérarchie originale comporte quatre classes (Classe A à Classe D) et
la nouvelle hiérarchie comporte quatre classes supplémentaires (Classe E,
25 Classe F, Classe A2, Classe C2) dont certaines (Classe A2 et Classe C2) sont
insérées dans la hiérarchie originale.

Dans ce cas, une table de correspondance peut être établie.

Comme illustré figure 2, cette table de correspondance peut, par exemple, donner, pour chaque classe de l'application originale, l'index I1, I3, I4, I6 de la classe correspondante dans la nouvelle application.

5 L'exemple suivant concerne un cas particulier dans lequel la table supplémentaire n'a pas besoin d'être incluse dans le fichier de chargement. Ceci est possible si :

- La hiérarchie des classes de l'application originale est conservée inchangée
10 dans la nouvelle version de l'application (aucune classe n'est insérée dans la hiérarchie).
- Les classes de l'application originale sont définies en premier dans le nouveau fichier de chargement, et dans le même ordre que dans le fichier de chargement original.
15

Les figures 3a et 3b montrent une hiérarchie avec la propriété recherchée dans laquelle la nouvelle hiérarchie (figure 3b) inclut les classes E et F sans insertion dans la hiérarchie originale (figure 3a) des classes A à D.

20 La figure 4 montre les tables de classes correspondant à cette hiérarchie, et respectant la propriété d'ordre indiquée ci-dessus. On voit alors que la table de correspondance est triviale et n'est pas nécessaire.

25 Le fichier doit également contenir une table de correspondance entre les champs statiques de l'application originale et ceux de la nouvelle version de l'application. Cette table est similaire à la précédente, mais elle est cette fois indexée par les identifiants des nouveaux champs statiques ; les éléments de la table qui correspondent à des nouveaux champs contiennent un identifiant invalide (I0 dans l'exemple), et les autres éléments contiennent l'identifiant du même champ dans la version originale.
30

La figure 5 montre un exemple d'une telle table comprenant :

- une table originale comportant les champs A.champ1, A.champ2,
5 C.champ1, A.champ3,
- une nouvelle table comportant les champs A.champ2, A.champ1,
C.champ1, C.champ2, A.champ3, F.champ1,
- 10 - une table de correspondance dans laquelle le champ A.champ1 est indexé par l'identifiant I1, A.champ2 est indexé par l'identifiant I2, C.champ1 est indexé par l'identifiant I3, A.champ3 est indexé par l'identifiant I4 ; les nouveaux champs C.champ2 et F.champ1 contiennent un identifiant invalide = I0.

15

Une fois le fichier approprié généré, il faut le charger dans la carte. On peut supposer que l'état de la carte avant chargement est comme indiqué en figure 6 : avec un objet Obj1 en ClasseA et un objet Obj2 en ClasseB de l'application App1.

20

Le chargement s'effectue en utilisant la procédure d'édition de liens normale du système.

25 La figure 7 montre l'état de la carte après le chargement de la nouvelle application. La nouvelle version de l'application App1' est chargée, mais les objets Obj1, Obj2 de l'application pointent toujours vers l'ancienne version (ClasseA, ClasseB de l'application App1).

30 Une des phases du chargement est l'initialisation des champs statiques. La procédure standard d'initialisation est appliquée, sauf pour les champs qui sont hérités de l'application originale. Pour ces champs, la valeur initiale définie

dans la nouvelle version de l'application est ignorée, et l'ancienne valeur est copiée.

5 L'étape suivante consiste donc à modifier les liens des objets pour qu'ils pointent vers les nouvelles classes. Il faut alors parcourir tous les objets de l'application et utiliser la table de correspondance entre anciennes et nouvelles classes pour identifier la nouvelle classe de l'objet.

10 Le résultat est montré figure 8, où les objets Obj1, Obj2 pointent sur les nouvelles classes (ClasseA', ClasseB').

15 Lors de cette étape, il se peut que les objets doivent être modifiés, si de nouveaux champs ont été ajoutés à leur classe. Dans ce cas, un nouvel objet (avec les nouveaux champs) est alloué, les valeurs des anciens champs sont copiées de l'ancienne version de l'objet, et les valeurs des nouveaux champs sont initialisées à leur valeur par défaut (0 pour les entiers, "false" pour les booléens, et "null" pour les pointeurs). Les références vers l'ancien objet sont ensuite mises à jour en utilisant des techniques classiquement implémentées dans les récupérateurs de mémoire.

20

La dernière étape consiste à laisser l'application faire toutes les opérations nécessaires à la mise à jour des données et, en particulier, de procéder à l'initialisation des nouveaux champs (champs statiques, ou champs insérés dans des objets). Les applets qui ont besoin d'effectuer une mise à jour doivent 25 implémenter une interface spécifique, dans laquelle une méthode est définie. La mise à jour consiste donc à parcourir la table des applets enregistrées dans le système et, pour chaque applet qui est une instance d'une classe définie dans le package mis à jour et qu'il implemente l'interface de mise à jour, d'invoquer la méthode avec les paramètres appropriés.

Revendications

1. Procédé pour le chargement sur un dispositif informatique d'une nouvelle version d'une application dans un langage de programmation à objets
5 et permettant, notamment, l'introduction de classes supplémentaires, la modification de la hiérarchie de classes et la définition de champs et de méthodes supplémentaires,
caractérisé en ce qu'il consiste à :

- 10 - calculer préalablement à ce chargement une information permettant d'établir la correspondance entre les classes de l'ancienne version de l'application et les classes de la nouvelle version de l'application ;
- calculer préalablement à ce chargement une information permettant d'établir la correspondance entre les identificateurs des champs statiques de l'ancienne version de l'application des identificateurs des champs statiques de la nouvelle version de l'application ;
15 - associer lesdites informations de correspondance à la nouvelle version de l'application chargée sur le dispositif ;
- utiliser lesdites informations de correspondance pour modifier les objets de façon à ce qu'ils pointent vers les classes de la nouvelle version de l'application et qu'ils utilisent les nouveaux identificateurs des champs statiques de la nouvelle version de l'application.

2. Procédé selon la revendication 1,
25 caractérisé en ce que lesdites informations de correspondance sont des tables de correspondance.

3. Procédé selon l'une des revendications 1 et 2,
caractérisé en ce que lesdites informations de correspondance sont omises
30 dans le cas où les modifications des objets ne sont pas nécessaires quand aucune classe supplémentaire n'est ajoutée dans la nouvelle version de

l'application ou quand les classes ajoutées ne modifient pas la hiérarchie de classes.

4. Procédé selon l'une des revendications précédentes,
5 caractérisé en ce qu'il comprend l'exécution de méthodes de mise à jour des données de l'application après l'installation de la nouvelle version de l'application.

5. Procédé selon l'une des revendications précédentes,
10 caractérisé en ce que ledit dispositif informatique est une carte à puce.

6. Procédé selon l'une des revendications précédentes,
caractérisé en ce que ledit langage de programmation est le langage « Java Card ».

Fig 1a

Héritage originale

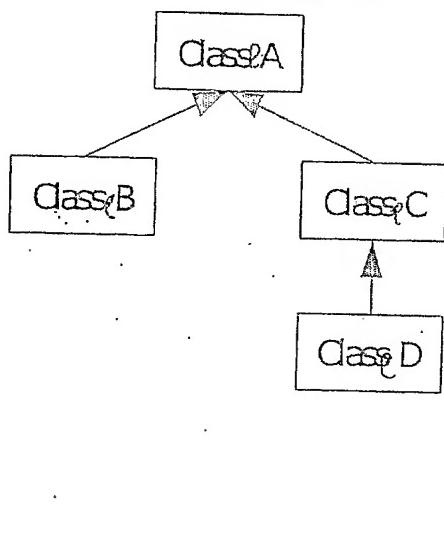


Fig 1b

Nouvelle hiérarchie

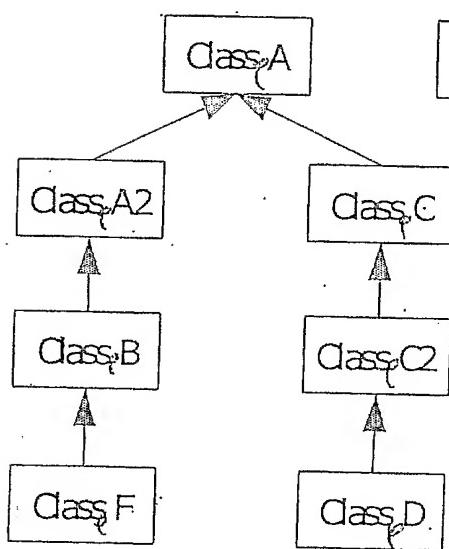


Fig 2

Table originale

Class A
Class B
Class C
Class D

Table de correspondance

I1
I3
I4
I6

Nouvelle table

Class A
Class A2
Class B
Class C
Class C2
Class D
Class E
Class F

1/3

Figure 1a

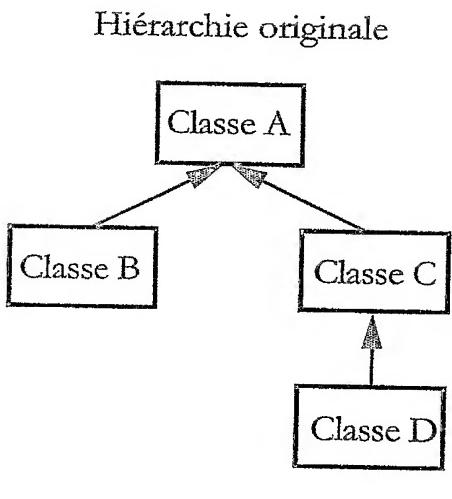


Figure 1b

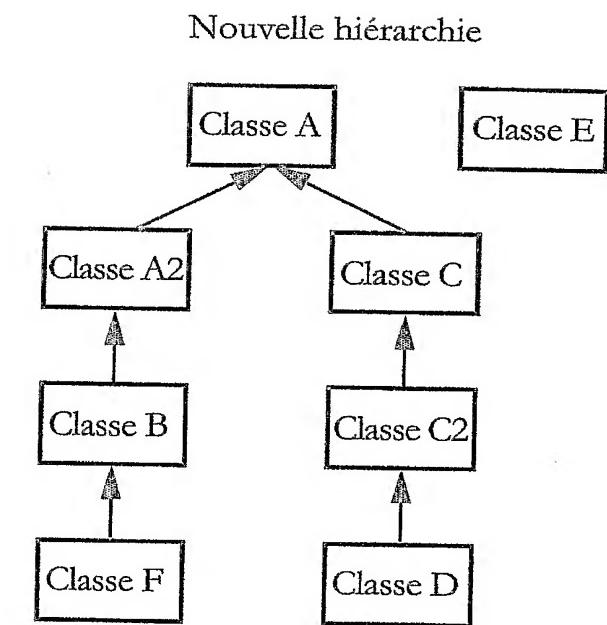


Figure 2

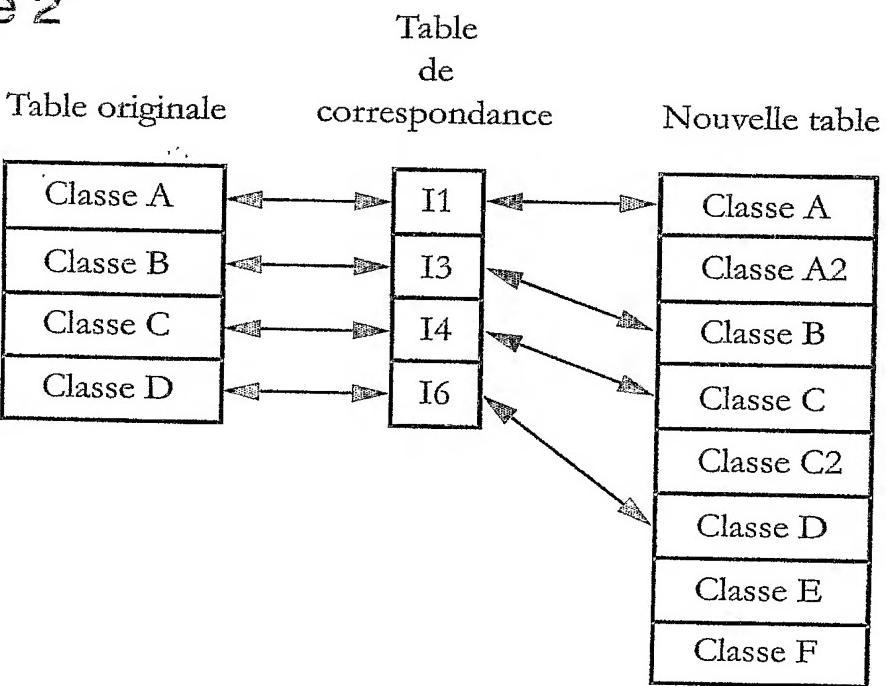


Fig 3a

Hiérarchie originale

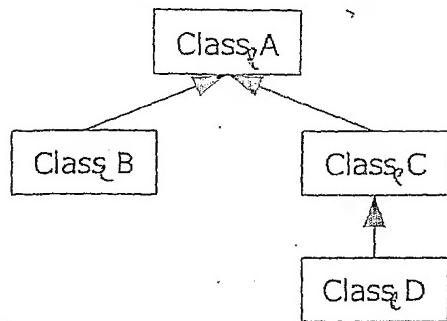


Fig 3b

Nouvelle hiérarchie

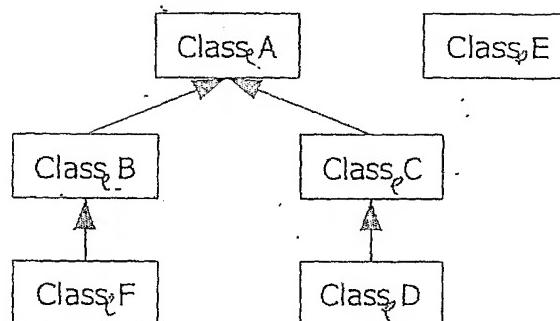


Fig 4

Table de correspondance

Table originale

Class A
Class B
Class C
Class D

Table de correspondance

I1
I2
I3
I4

Nouvelle table

Class A
Class B
Class C
Class D
Class E
Class F

Fig 5

Table de correspondance

Table originale

A.champ1
A.champ2
C.champ1
A.champ3

Table de correspondance

I2
I1
I3
I0
I4
I0

Nouvelle table

A.champ2
A.champ1
C.champ1
C.champ2
A.champ3
F.champ1

2/3

Figure 3a

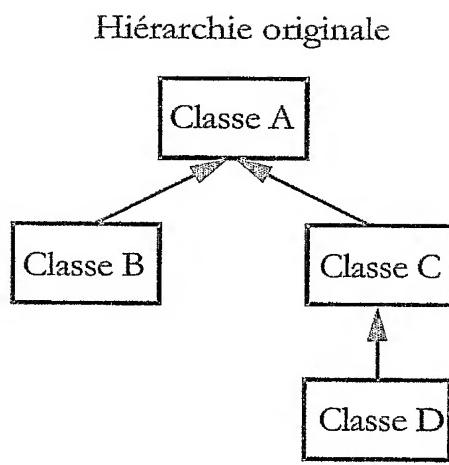


Figure 3b

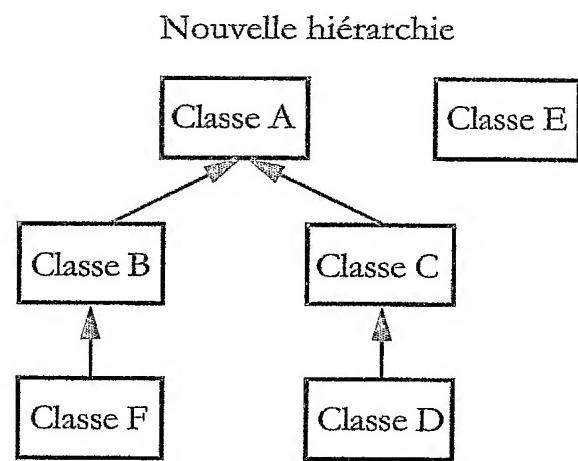


Figure 4

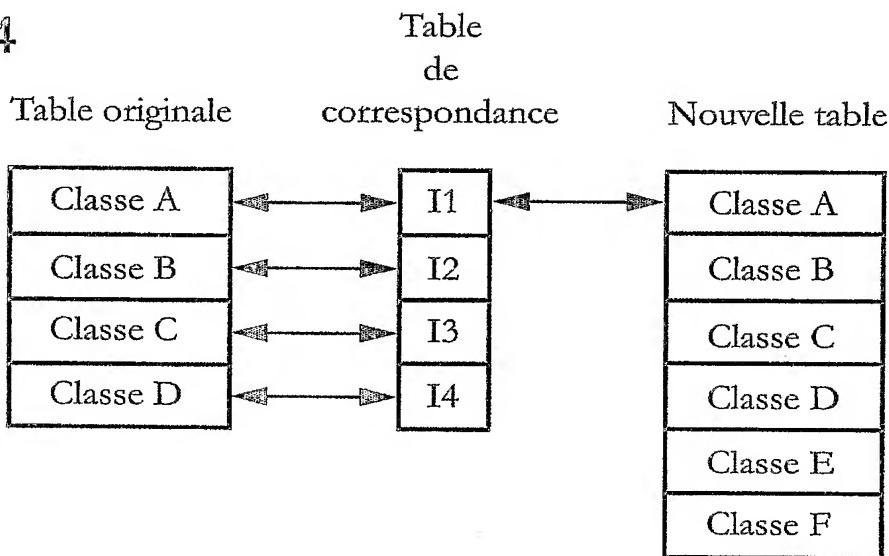


Figure 5

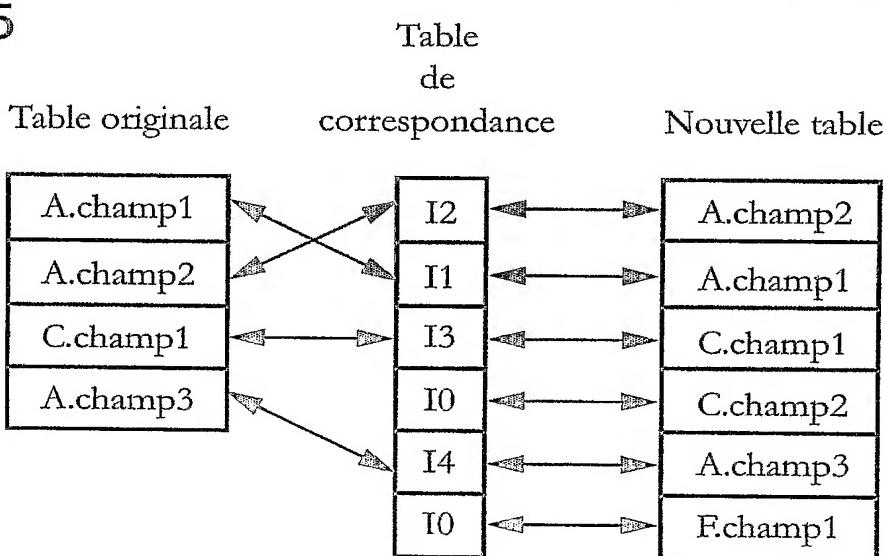


Fig 6

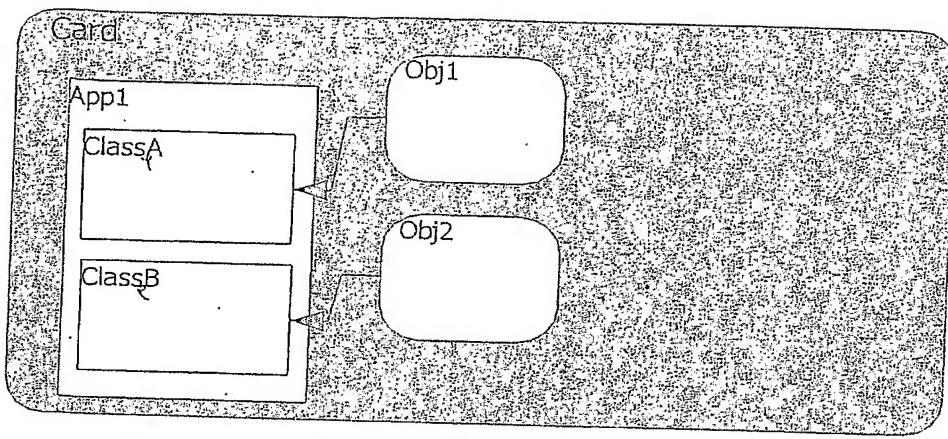


Fig 7

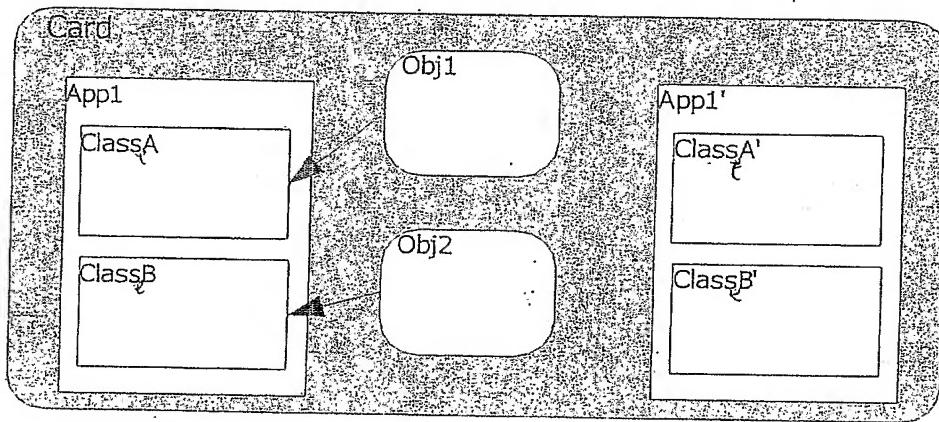
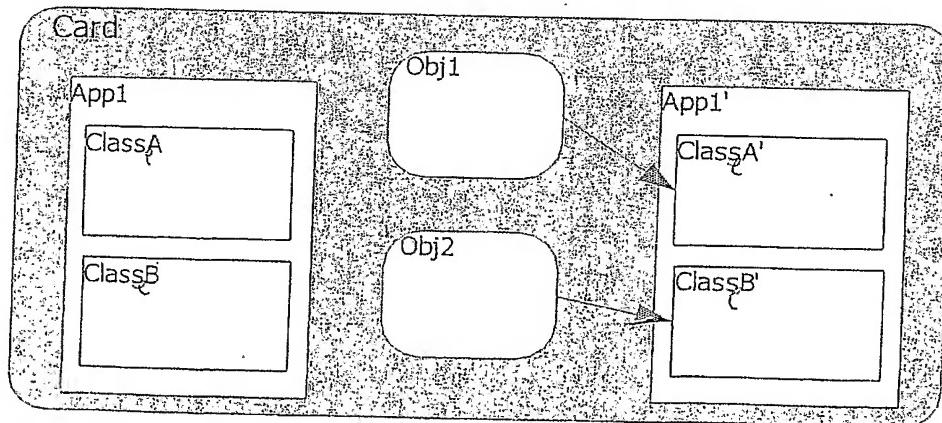


Fig 8



3/3

Figure 6

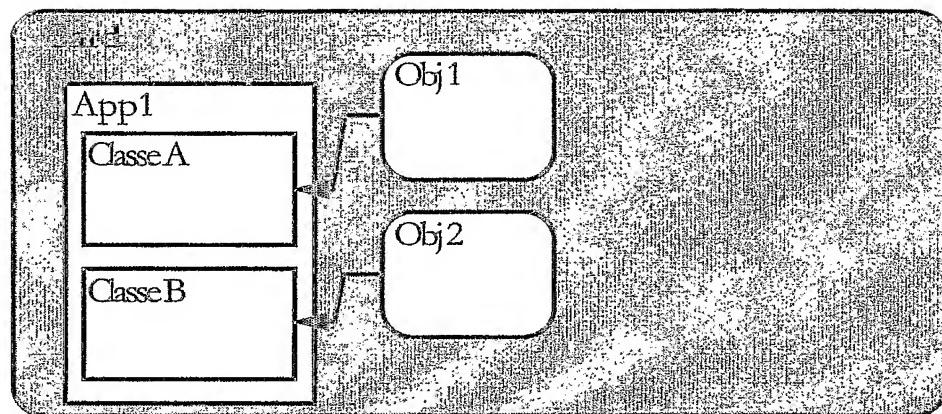


Figure 7

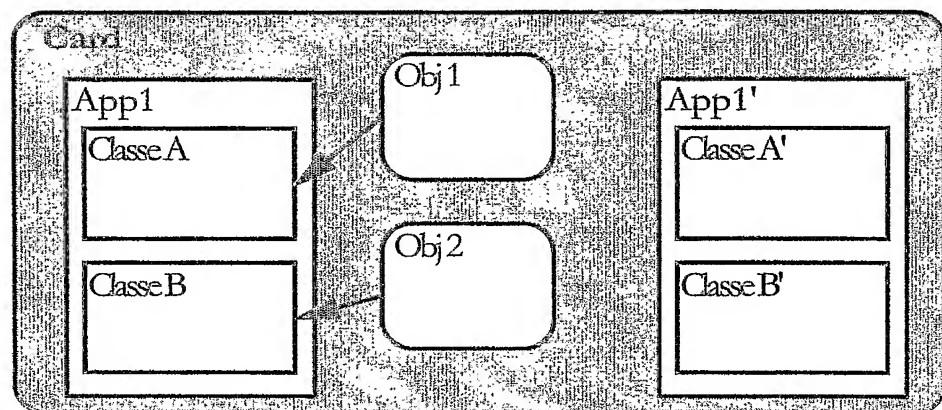
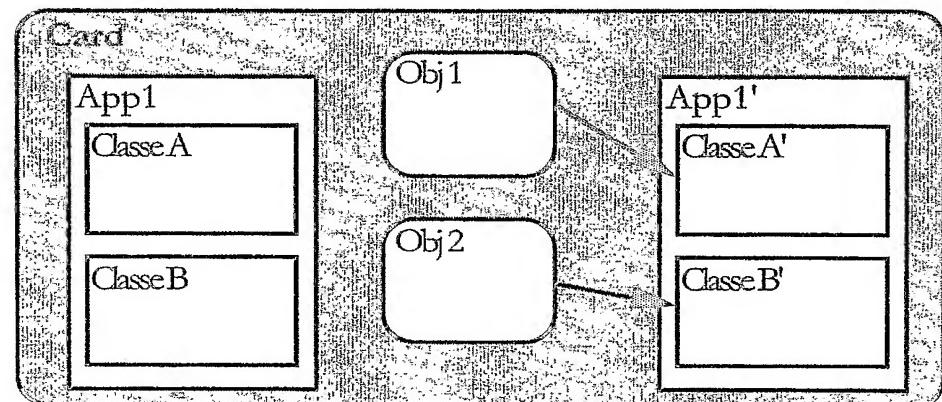


Figure 8



DÉPARTEMENT DES BREVETS

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 33 (1) 53 04 53 04 Télécopie : 33 (1) 42 94 86 54

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ

Code de la propriété intellectuelle - Livre VI

DÉSIGNATION D'INVENTEUR(S) Page N° 1.../1...

(À fournir dans le cas où les demandeurs et les inventeurs ne sont pas les mêmes personnes)

Cet imprimé est à remplir lisiblement à l'encre noire

DB 113 @ W / 270601

INV

Vos références pour ce dossier (facultatif)	TRUSB0020	
N° D'ENREGISTREMENT NATIONAL		
TITRE DE L'INVENTION (200 caractères ou espaces maximum)		
PROCEDE DE MISE A JOUR D'APPLICATIONS POUR CARTE A PUCE,		
LE(S) DEMANDEUR(S) :		
CABINET MOUTARD - 35 rue de la Paroisse - 78 000 Versailles - agissant en qualité de mandataire auprès de :		
TRUSTED LOGIC 5 rue du Bailliage 78 000 VERSAILLES		
DESIGNE(NT) EN TANT QU'INVENTEUR(S) :		
<input checked="" type="checkbox"/> Nom	VETILLARD	
Prénoms	Eric	
Adresse	Rue	Résidence Garbejaire 2, Bât 1 1 Passage des Pignes
	Code postal et ville	10 6 5 6 0 VALBONNE
Société d'appartenance (facultatif)		
<input checked="" type="checkbox"/> Nom		
Prénoms		
Adresse	Rue	
	Code postal et ville	
Société d'appartenance (facultatif)		
<input checked="" type="checkbox"/> Nom		
Prénoms		
Adresse	Rue	
	Code postal et ville	
Société d'appartenance (facultatif)		

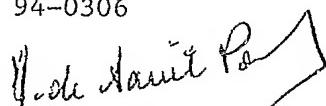
S'il y a plus de trois inventeurs, utilisez plusieurs formulaires. Indiquez en haut à droite le N° de la page suivie du nombre de pages.

DATE ET SIGNATURE(S)**DU (DES) DEMANDEUR(S)****OU DU MANDATAIRE**

(Nom et qualité du signataire)

23 décembre 2003 - N° 94-0306

A. de Saint Palais



PCT/FR2004/003353

